PLEASE ENTER AMENDMENT to ClAIMS AND SPECIFICATIONS included herein.

VAT

6/6/08

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Kenneth W. COWAN, et al.		
Serial No.:	10/754,017	Examiner:	VU, Tuan A.
Filed:	January 7, 2004	Art Unit:	2193
For:	TECHNIQUES FOR SOFTWARE CONFIGURATION TRACKING	Atty. Docket:	NUM-02402

Certificate of Mailing

I hereby certify that the foregoing document is being deposited with the United States Postal Service, postage prepaid, first class mail, in an envelope addressed to Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on June 6, 2008.

_____

Sandra Pires

AMENDMENT AND RESPONSE TO OFFICE ACTION
SUPPLEMENTAL

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This paper is being provided in response to the Office Action dated January 2, 2008, for the above-captioned U.S. patent application and SUPPLEMENTS the Amendment and Response submitted by Applicants on March 28, 2008.

Amendments to the Specification are listed beginning on page 2 of this paper.

Amendments to the Claims are listed beginning on page 3 of this paper.

Remarks begin on page 19 of this paper.

It is not believed that extensions of time or fees for net addition of claims are required, beyond those which may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary, then such extensions of time are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required for consideration of this paper (including fees for net addition of claims) are authorized to be charged in two originally-executed copies of a Transmittal Letter filed herewith.

AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 7, line 13, as follows:

The monitor process 30 may be a machine executable program written in any one of a variety of commercially available programming languages, such as C++. An example of a monitor process is included in the commercially available implementation of NuMega BoundsChecker. Generally, the monitor process 30 gathers information and data in accordance with the execution of the program under test 44 which in this example may be referred to as test.exe. The machine executable program 44 being monitored in this particular example may be written in one of a variety of conventional, commercially available languages, such as C, and C++, for which instrumentation may be provided. One technique of how instrumentation of a program may be performed is described in U.S. Patent Application No. 08/916,125, entitled "IR Code Instrumentation", now issued as U.S. Patent No. 5,987,249.

AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions and listing of claims in the above-referenced application.

LISTING OF CLAIMS:

1. (Currently amended) A computer implemented method for automatically tracking build information to determine a volatility metric between a first build of a first version of software and a second build of a second version of software, comprising:

extracting the second build ~~information~~ by processing, for ~~each of one or more builds~~ the second version of the software, one or more software modules produced using a compilation process resulting in said one or more software modules of the second version of software, wherein said second build ~~information extracted~~ includes at least one of: a name field, a build number, a date and time identifier, a number of functions and modification information for the functions ~~software module information about~~ being associated with at least one software module ~~produced as an output of a~~ as a result from said compilation process ~~for each of said one or more builds~~;

following extracting the second build, registering said ~~one or more builds~~ second build by storing said second build ~~information~~ corresponding to ~~each of said one or more builds~~ said second version of software in a database;

executing said second version of software;

automatically determining ~~software module information about software being tested, wherein said software module information is gathered from one or more software modules~~ runtime data during execution of said second version of the software ~~being tested~~, wherein the runtime data corresponds to information extracted and registered for the second version of the software;

performing a query of the database to retrieve the second build based on the runtime data and to retrieve ~~first build information for a~~ the first build ~~of the one or more builds in the database~~; and

after performing the query of the database to retrieve the first build and the second build, determining a volatility metric of code change that has occurred between the first and second versions of the software ~~modules in both said first build information and said software module information of the software being tested~~, wherein ~~said~~ determining the volatility metric includes matching portions of the first build ~~information~~ to corresponding portions of said ~~software module information~~ second build and determining at least one of: a date and time difference, a build number difference, a number of functions added, a number of functions removed and a number of functions modified in a software module of the ~~software module information~~ second version of the software in comparison to a software module of the first ~~build information~~ version of the software.

2. (Cancelled)

3. (Previously presented) The method of Claim 1, wherein said database that includes said build information is an object database, and the method further comprises:

creating and storing one or more objects corresponding to each of said builds; and

creating and storing one or more objects corresponding to software modules included in each of the builds.

4. (Currently amended) The method of Claim 3, further including:

creating and storing a session object corresponding to a test session of said software ~~being tested~~;

creating and storing one or more objects corresponding to software modules describing said software module information;

automatically determining a previously created build object corresponding to one of said one or more builds previously registered; and

storing an address of said previously created build object in said session object.

5. -6. (Cancelled)

7. (Currently Amended) The method of Claim [[6]]1, wherein subroutine calls associated with an operating system executing in said computer system are used in said gathering runtime information.

8. (Currently Amended) The method of Claim 1, further comprising:

~~using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and~~

creating and storing a bug report object corresponding to ~~said~~ a bug report, said bug report object including an address associated with said second build.

9. (Original) The method of Claim 8, further including:

submitting said bug report included in a formatted electronic message;

interpreting data included in said formatted electronic message to enable said determining of said second build.

10. (Original) The method of Claim 9, further including:

creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

associating said other build object with said bug report object in said database.

11. (Currently Amended) The method of Claim 1, wherein said automatically determining said ~~first build~~ runtime data further comprises:

determining said ~~first~~ second build for which a matching build is being determined from said one or more builds registered;

determining a candidate list including one or more builds having at least one software module in common with said ~~first~~ second build;

for each build included in said candidate list, determining if software modules included in said each build match software modules included in said ~~first~~ second build;

for each build included in said candidate list, if there are no software modules included in said each build that have a module name and associated attributes matching a software module included in said ~~first~~ second build, determining that said each build is not a match for said ~~first~~ second build; and

for each build included in said candidate list, if a first of said software modules included in said each build has a module name that matches a software module included in said ~~first~~ second build but attributes associated with said software module do not match said software module included in said ~~first~~ second build, determining that said each build is not a match for said ~~first~~ second build.

12. (Currently Amended) The method of Claim 11, further including:

for each build included in said candidate list, determining a number of matches for software modules included in said each build having a matching module name and attributes of a software module included in said ~~first~~ second build;

determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for software modules included in said each build with software modules included in said ~~first~~ second build, each software module included in said each build having a corresponding matching software module included in said ~~first~~ second build, said name and associated attributes of said each software module matching said matching software module included in said ~~first~~ second build; and

determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one software module having a module name that does not have a matching software module included in said ~~first~~ second build, said each build including at least one software module having a module name and associated attributes matching another software module included in said ~~first~~ second build, wherein said each build has an associated number of matches.

13. (Currently Amended) The method of Claim 12, further including:

if said valid list is empty, for each project, adding a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

performing an alternative action if said valid list is empty;

if there is only one build included in said valid list, determining that said one build matches said ~~first~~ second build; and

if there is more than one build included in said valid list, selecting one of the more than one builds included in said valid list as being the build matching said ~~first~~ second build.

14. (Original) The method of Claim 13, further including:

selecting one or more build associated with a software project.

15. (Original) The method of Claim 13, further including:

determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

16. - 21. (Cancelled)

22. (Currently Amended) A computer readable medium comprising machine executable code stored thereon for automatically tracking build information to determine a volatility metric between a first build of a first version of software and a second build of a second version of software, the computer readable medium comprising:

machine executable code for extracting the second build ~~information~~ by processing, for ~~each of one or more builds~~ the second version of the software, one or more software modules produced using a compilation process resulting in said one or more software modules of the second version of software, wherein said second build ~~information extracted~~ includes at least one of: a name field, a build number, a date and time identifier, a number of functions and modification information for the functions ~~software module information about~~ being associated with at least one software module ~~produced as an output of a~~ as a result from said compilation process ~~for each of said one or more builds~~;

machine executable code for registering said ~~one or more builds~~ second build following extracting the second build by storing said second build ~~information~~ corresponding to ~~each of said one or more builds~~ said second version of software in a database;

machine executable code for automatically determining ~~software module information about software being tested, wherein said software module information is gathered from one or more software modules~~ runtime data during execution of said second version of the software ~~being tested~~, wherein the runtime data corresponds to information extracted and registered for the second version of the software; ~~and~~

machine executable code for performing a query of the database to retrieve the second build based on the runtime data and to retrieve the first build; and

machine executable code for ~~determining a first of said one or more builds included in the database which corresponds to said software module information about the software being tested, the determining including~~ determining a volatility metric of code change that has occurred between the first and second versions of the software ~~modules in both said first build information and said software module information of the software being tested~~ after performing the query of the database to retrieve the first build and the second build, , wherein ~~said~~ determining the volatility metric includes matching portions of the first build ~~information~~ to corresponding portions of said ~~software module information~~ second build and determining at least one of: a date and time difference, a build number difference, a number of functions added, a number of functions removed and a number of functions modified in a software module of the ~~software module information~~ second version of the software in comparison to a software module of the first ~~build information~~ version of the software.

23. (Cancelled)

24. (Previously presented) The computer readable medium of Claim 22, wherein said database that includes said build information is an object database, and the computer program product further comprises:

machine executable code for creating and storing one or more objects corresponding to each of said builds; and

machine executable code for creating and storing one or more objects corresponding to software modules included in each of the builds.

25. (Currently amended) The computer readable medium of Claim 24, further including:

machine executable code for creating and storing a session object corresponding to a test session of said software ~~being tested~~ ;

machine executable code for creating and storing one or more objects corresponding to software modules describing said software module information;

machine executable code for determining automatically a previously created build object corresponding to one of said one or more builds previously registered; and

machine executable code for storing an address of said previously created build object in said session object.

26. -27. (Cancelled)

28. (Currently Amended) The computer readable medium of Claim [[27]]22, wherein machine executable code associated with an operating system is used in gathering runtime information.

29. (Currently Amended) The computer readable medium of Claim 22, further including:

~~machine executable code for using said build information in said database to automatically determine a second of said one or more builds corresponding to software module information included in a bug report; and~~

machine executable code for creating and storing a bug report object corresponding to ~~said~~ a bug report, said bug report object including an address associated with said second build.

30. (Previously presented) The computer readable medium of Claim 29, further including:

machine executable code for submitting a bug report included in a formatted electronic message; and

machine executable code for interpreting data included in said formatted electronic message to enable determination of said second build.

31. (Previously presented) The computer readable medium of Claim 30, further including:

machine executable code for creating and storing another build object corresponding to a build in which said bug report is identified as being corrected; and

machine executable code for associating said other build object with said bug report object in said database.

32. (Currently Amended) The computer readable medium of Claim 22, wherein said machine executable code for automatically determining said ~~first build~~ runtime data further comprises:

machine executable code for determining said ~~first~~ second build for which a matching build is being determined from said one or more builds registered;

machine executable code for determining a candidate list including one or more builds having at least one software module in common with said ~~first~~ second build;

machine executable code for determining, for each build included in said candidate list, if software modules included in said each build match software modules included in said ~~first~~ second build;

machine executable code for determining, for each build included in said candidate list, if there are no software modules included in said each build that have a module name and associated attributes matching a software module included in said ~~first~~ second build, determining that said each build is not a match for said ~~first~~ second build; and

machine executable code for determining, for each build included in said candidate list, that said each build is not a match for said ~~first~~ second build if a first of said software modules included in said each build has a module name that matches a software module included in said ~~first~~ second build but attributes associated with said software module do not match said software module included in said ~~first~~ second build.

33. (Currently Amended) The computer readable medium of Claim 22, further including:

machine executable code for determining, for each build included in said candidate list, a number of matches for software modules included in said each build having a matching module name and attributes of a software module included in said ~~first~~ second build;

machine executable code for determining a valid list of one or more matching builds, each of said builds included in said valid list having a full match for software modules included in said each build with software modules included in said ~~first~~ second build, each software module included in said each build having a corresponding matching software module included in said ~~first~~ second build, said name and associated attributes of said each software module matching said matching software module included in said ~~first~~ second build; and

machine executable code for determining a maybe list of one or more builds, each of said one or more builds included in said maybe list having at least one software module having a module name that does not have a matching software module included in said ~~first~~ second build, said each build including at least one software module having a module name and associated attributes matching another software module included in said ~~first~~ second build, wherein said each build has an associated number of matches.

34. (Currently Amended) The computer readable medium of Claim 33, further including:

machine executable code for adding, if said valid list is empty, for each project, a build from said maybe list to said valid list in which the build added has a maximum number of matches of builds associated with said each project, a project having one or more associated builds;

machine executable code for performing an alternative action if said valid list is empty;

machine executable code for determining that said one build matches said ~~first~~ second build if there is only one build included in said valid list; and

machine executable code for selecting one of the more than one builds included in said valid list as being the build matching said ~~first~~ second build if there is more than one build included in said valid list.

35. (Previously presented) The computer readable medium of Claim 34, further including:

machine executable code for selecting one or more build associated with a software project.

36. (Previously presented) The computer readable medium of Claim 34, further including:

machine executable code for determining a matching build in accordance with a predetermined build selected from a list of more than one build previously registered.

37. - 42. (Cancelled)

REMARKS

This paper is being provided in response to the Office Action dated January 2, 2008, for the above-referenced application and SUPPLEMENTS the Amendment and Response submitted on March 28, 2008 by Applicants.

Applicants thank Examiner Twan Vu for the recent telephone discussions with Applicants' undersigned representative. Applicants have amended the claims herein in accordance with the telephone discussions

With respect to the provisional rejections of claims 17, 38, 1, 20, 22, and 41 on the grounds of nonstatutory obviousness-type double patenting over claims 18, 39, 11, and 33 of copending application No. 09/547,550, Applicants note that a Terminal Disclaimer was filed by Applicants on May 30, 2008.

Based on the above, Applicant respectfully requests that the Examiner reconsider and withdraw all outstanding rejections and objections. Favorable consideration and allowance are earnestly solicited. Should there be any questions after reviewing this paper, the Examiner is invited to contact the undersigned at 508-898-8603.

Respectfully submitted,
MUIRHEAD AND SATURNELLI, LLC

Date: June 6, 2008

_____
Donald W. Muirhead
Reg. No. 33,978

Muirhead and Saturnelli, LLC
200 Friberg Parkway, Suite 1001
Westborough, MA 01581
Tel: (508) 898-8601
Fax: (508) 898-8602